7 Quasi-Newton (secant) methods

Quasi-Newton methods are probably the most popular general-purpose algorithms for unconstrained optimization. The basic idea behind quasi-Newton methods is quite simple. A typical iteration of the method is

$$x^{k+1} = x^k + \alpha^k d^k, \text{ where } d^k = -\mathrm{B}^k \nabla f(x^k),$$

where B^k is a positive definite matrix (which is adjusted from iteration to iteration) chosen so that the directions d^k tend to approximate Newton's direction. The stepsize α^k is usually chosen by a line search.

Many quasi-Newton methods are advantageous due to their fast convergence and absence of second-order derivative computation. We will compare them to the conjugate gradient methods later on.

1 The Broyden family

Of course, what makes a quasi-Newton method work is the choice of the matrix B^k at each iteration. The important idea behind the methods is that two successive iterates x^k and x^{k+1} together with the gradients $\nabla f(x^k)$ and $\nabla f(x^{k+1})$ contain curvature (i.e., Hessian) information, in particular,

$$(\nabla f(x^{k+1}) - \nabla f(x^k)) \approx H(x^{k+1})(x^{k+1} - x^k)$$

(observe that the above approximation is an *equality* when the function in question is quadratic!). Therefore, at every iteration we would like to choose B^{k+1} to satisfy

$$\mathrm{B}^{k+1} q^k = p^k, \text{ where } p^k = x^{k+1} - x^k, \; q^k = \nabla f(x^{k+1}) - \nabla f(x^k). \tag{21}$$

Equation (21) is known as the quasi-Newton condition, or the secant equation.

Suppose that at every iteration we update the matrix B^{k+1} by taking the matrix B^k and adding a "correction" term C^k. Then the secant equation becomes

$$(\mathrm{B}^k + C^k) q^k = p^k \;\Rightarrow\; C^k q^k = p^k - \mathrm{B}^k q^k. \tag{22}$$

Note that equation (22) leaves us a lot of flexibility in selecting the correction matrix C^k. The most popular update methods come from the following (parametric) family of matrices (the superscript k is omitted in most of the following formulas for simplicity, here $\mathrm{B} = \mathrm{B}^k$):

$$C^{\mathrm{B}}(\xi) = \frac{pp^t}{p^t q} - \frac{\mathrm{B}qq^t\mathrm{B}}{q^t \mathrm{B} q} + \xi \tau v v^t, \text{ where } v = \frac{p}{p^t q} - \frac{\mathrm{B}q}{\tau}, \; \tau = q^t \mathrm{B} q \tag{23}$$

(it is not hard to verify that these updates indeed satisfy the secant equation).

The choice of the scalar $\xi \in [0, 1]$, which parameterizes the family of matrices C, gives rise to several popular choices of updates. In particular:

- Setting $\xi = 0$ at each iteration, we obtain the so-called DFP (Davidson-Fletcher-Powell) update:WE USE B=D
$$C^{\mathrm{DFP}} = C^{\mathrm{B}}(0) = \frac{pp^t}{p^t q} - \frac{Dqq^t D}{q^t Dq},$$
which is historically the first quasi-Newton method

- Setting $\xi = 1$ at each iteration, we obtain the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update:
$$C^{\mathrm{BFGS}} = C^{\mathrm{B}}(1) = \frac{pp^t}{p^t q}\left[1 + \frac{q^t Dq}{p^t q}\right] - \frac{Dqp^t + pq^t D}{p^t q}.$$
The resulting method has been shown to be superior to other updating schemes in its overall performance.

- A general member of the Broyden family (23) can therefore be written as a convex combination of the two above updates:
$$C^{\mathrm{B}}(\xi) = (1-\xi)C^{\mathrm{DFP}} + \xi C^{\mathrm{BFGS}}$$

The following two results demonstrate that quasi-Newton methods generate descent search directions (as long as exact line searches are performed, and the initial approximation D^1 is positive definite), and, when applied to a quadratic function, result in conjugate direction methods.

Proposition 33 (Bertsekas, Prop. 1.7.1) *If D^k is positive definite and the stepsize α^k is chosen so that x^{k+1} satisfies*
$$(\nabla f(x^k) - \nabla f(x^{k+1})^t d^k < 0,$$
then D^{k+1} given by (23) is positive definite (and hence d^{k+1} is a descent direction).

Note that if exact line search is performed, $\nabla f(x^{k+1})^t d^k = 0$, so that condition above is satisfied.

Proposition 34 (Bertsekas, Prop. 1.7.2) *If the quasi-Newton method with matrices D^k generated by (23) is applied to minimization of a positive-definite quadratic function $f(x) = \frac{1}{2}x^t Qx - q^t x$, then the vectors d^i, $i = 1, \ldots, n$ are Q-conjugate, and $D^{n+1} = Q^{-1}$.*

Note that the above proposition indicates that the algorithm not only generates conjugate directions, but also simultaneously constructs the matrix Q^{-1}. (A side note: if $D^1 = I$, the method actually coincides with the conjugate gradient algorithm).

7.2 BFGS method

An alternative to maintaining the matrix D^k above which approximates the inverse of the Hessian, one could maintain a positive definite matrix B^k which would approximate the Hessian itself. Viewed from this perspective, the secant equation can be written as
$$q^k = B^{k+1}p^k = (B^k + \tilde{C}^k)p^k \;\Rightarrow\; \tilde{C}^k p^k = q^k - B^k p^k,$$
where $\tilde{C}$ is the "correction" matrix in this setting. Analogously to $C^{\mathrm{B}}(\xi)$, one can construct a parametric family of update matrices
$$\tilde{C}(\phi) = \frac{qq^t}{q^t p} - \frac{Bpp^t B}{p^t Bp} + \phi\tau vv^t, \text{ where } v = \frac{q}{q^t p} - \frac{Bp}{\tau},\ \tau = q^t Bq.$$

Using $\phi = 0$, we obtain the update used in the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method:

$$\tilde{C}^{\text{BFGS}} = \frac{qq^t}{q^t p} - \frac{Bpp^t B}{p^t Bp}.$$

(If it seems like we have two different objects referred to as "the BFGS," fear not – in fact, if $D = B^{-1}$, then $D + C^{\text{BFGS}} = (B + \tilde{C}^{\text{BFGS}})^{-1}$, so the two BFGS updates are consistent with each other).

When using this method, the search direction at each iteration has to be obtained by solving the system of equations

$$B^{k+1} d^{k+1} = -\nabla f(x^{k+1}).$$

It may appear that this will require a lot of computation, and approximating the inverse of the Hessian directly (as was done in the previous subsection) is a better approach. However, obtaining solutions to these systems is implemented quite efficiently by maintaining the so-called Cholesky factorization of the matrix $B^{k+1} = L\Lambda L^t$ (here L is a lower triangular matrix, and Λ is a diagonal matrix). This factorization is easy to implement and update and is numerically stable. In addition, if the line searches are not performed to sufficient precision (in particular, the resulting iterates do not satisfy the conditions of Proposition 33), the matrices D^k are not guaranteed to be positive definite. This would be fairly hard to detect, and can lead to bad choice of direction d^k. On the other hand, maintaining the Cholesky factorization of the matrix B^k immediately allows us to check the signs of eigenvalues of B^k (just look at the elements of Λ), and if needed add a correction term to maintain positive definiteness.

3 Comparison with conjugate gradient algorithms

We have seen that the quasi-Newton methods are closely related to conjugate directions and conjugate gradient algorithms when applied to quadratic functions. When minimizing a general non-quadratic function, quasi-Newton methods (BFGS in particular) typically perform better. Partially this is due to the fact that quasi-Newton methods, in addition to generating conjugate directions, also tend to approximate the Hessian matrix, and hence, close to the optimum, the directions they generate tend to approximate the Newton's direction. This observation holds true regardless of the starting matrix D^1, and hence it is typically unnecessary to restart quasi-Newton methods. Also, as is suggested by numerical evidence and (somewhat) by analysis, quasi-Newton methods tend to be less sensitive to the accuracy of line searches performed at each iteration.

On the other hand, compared to conjugate gradient methods, quasi-Newton methods require more storage space, and each iteration requires more computation. (In particular, they have to store the matrix D^k and compute $D^k \nabla f(x^k)$ at each iteration, while the number of function and gradient evaluations is the same as in conjugate gradient algorithm.) In general, both conjugate gradient and quasi-Newton methods require significantly less work per iteration than Newton's method (unless special structure of the problem allows for efficient direct computation of the Hessian and its inverse).

4 A final note

Quasi-Newton methods (typically with BFGS update of one form of another) are usually the algorithms of choice in unconstrained optimization software.

The optimization toolbox in MATLAB implements quite a few gradient descent methods in its function `fminunc`. The default method for small-to-medium size problems is the BFGS method (with update $\tilde{C}^{\text{BFGS}}$). The formula for gradient of the function f can be provided as a subroutine; if not available, the gradients will be approximated numerically.

The software allows you to change the algorithm used to DFP quasi-Newton method which approximates the inverse of the Hessian, or to steepest descent (the later, however, is "not recommended" by the manual).

For large-scaled problems, MATLAB implements a version of Newton's algorithms. An interesting aspect of the implementation (as is the case with many implementations of Newton's method) is that the computation of the direction $d = -H(x)^{-1}\nabla f(x)$ is often performed approximately by applying a few iterations of the conjugate gradient algorithm to solve the linear system $H(x)d = \nabla f(x)$.